

Topic 2: Molecular Dynamics of Lennard-Jones System (continued)

In the systolic algorithm, the N particles are decomposed into subsets which are assigned to each of the parallel processes. The algorithm is relatively straightforward to implement because each process deals with a fixed number of particles.

Spatial or Domain Decomposition

Another very useful concept is that of spatial or domain decomposition. Here, the physical volume is divided up into regions which are then assigned to different processes. Since the N particles move around in space, they must be moved from one process to another, and this makes the implementation a little more complicated.

Suppose that the system volume $V = L^3$ has the shape of a cube. There are essentially three different simple ways in which it can be decomposed into subregions:

1. It can be divided into $2^3=8, 3^3=27, 4^3=64, \dots$ cubes of side $L/2, L/3, L/4, \dots$. In this decomposition, each interior region has 26 neighboring regions—6 sharing a face, 12 sharing an edge, and 8 sharing a vertex. This is a full three-dimensional decomposition.
2. It can be divided into $2^2=4, 3^2=9, 4^2=16, \dots$ rectangular parallelepipeds or cuboids of length L and width = height = L/n , where $n = 2, 3, 4, \dots$. Each interior cuboid has 8 neighboring regions—4 sharing a face and 4 sharing an edge. This is an intermediate two-dimensional decomposition.

3. It can be divided into 2, 3, 4, ... cuboids of length = width = L and height = L/n , where $n = 2, 3, 4, \dots$. Each interior cuboid has 2 neighboring regions with which it shares a face. This is the simplest one-dimensional decomposition.

Virtual or Application Topology

In an MD simulation, particles can move from one spatial region to any of the neighboring regions at each time step. When this happens, the program must move the particles between the processes associated with these regions. These neighbor connections between the regions define a *virtual topology* or *application topology*.

For a given spatial decomposition, the virtual topology depends on the type of boundary conditions used in the physical problem. Two important examples:

- Closed boundary conditions: the particles cannot leave the physical volume. The surface regions have fewer neighbors than interior regions and must be treated differently from the interior regions.

In a one-dimensional decomposition, the regions have the topology of an open linear chain. The two- and three-dimensional topologies are those of a two- or three-dimensional Cartesian lattice.

- Periodic boundary conditions: the physical volume is imagined to be repeated periodically in one or more of the three spatial dimensions. With periodicity in all three dimensions all regions have the same number of neighbors.

In a one-dimensional decomposition, the regions have the topology of an closed linear ring. The two- and three-dimensional topologies are those of a two- or three-dimensional toroidal lattice.

Process Interconnection Topology

The processes (CPU's) on a parallel computer system are also connected to one another on a network. Not all network connections are identical: they might operate at very different speeds, for example in a network of dual-cpu computers connected via ethernet.

Parallel programming languages have facilities for mapping a virtual application topology to the interconnection topology of the parallel system.

MPI allows users to define a particular virtual application topology. This allows the MPI implementation to optimize the mapping of the virtual topology to the physical topology of the parallel machine. MPI has builtin functions to manipulate *Cartesian topologies*, which are the type most commonly encountered in applications:

- `MPI_Cart_create`: creates a new topology from a communicator. The number of dimensions, the number of processes in each dimension, whether or not the topology is periodic, and whether or not the processes can be reordered for better performance, can all be specified.

- `MPI_Cart_coords`: gets the cartesian coordinates of a process whose rank you specify.
- `MPI_Cart_shift`: gets the neighboring process in a specified direction and a specified number of steps away.